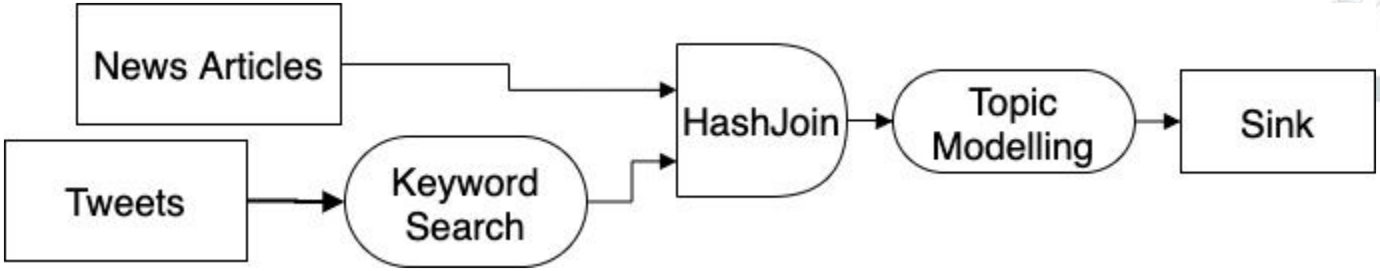
Amber : A Debuggable Dataflow System Based on the Actor Model

Avinash Kumar, Zuozhi Wang, Shengquan Ni and Chen Li

UC Irvine

Big data analytics workflows



News Articles

Tweets

Keyword Search

HashJoin

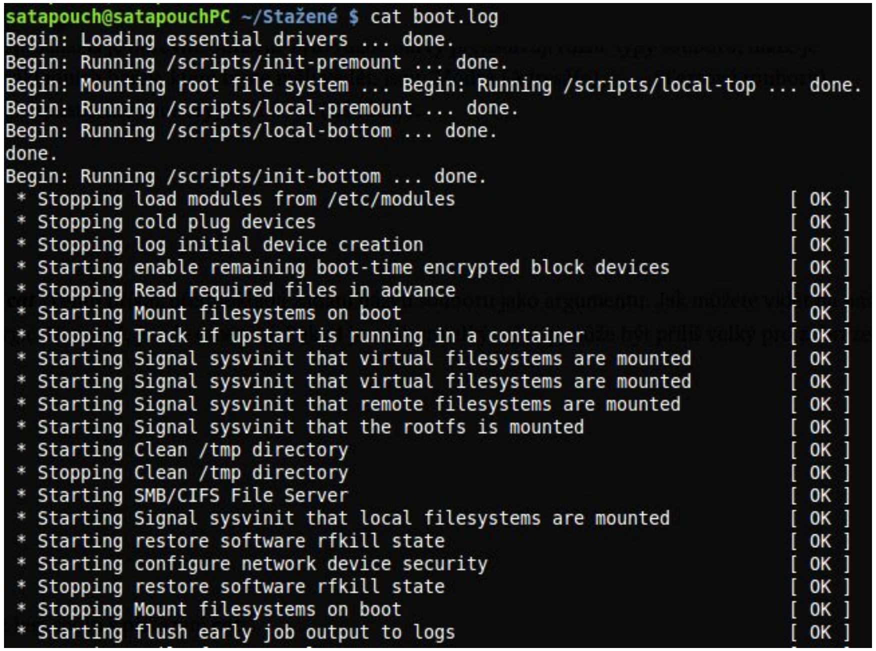
Topic Modelling

Sink

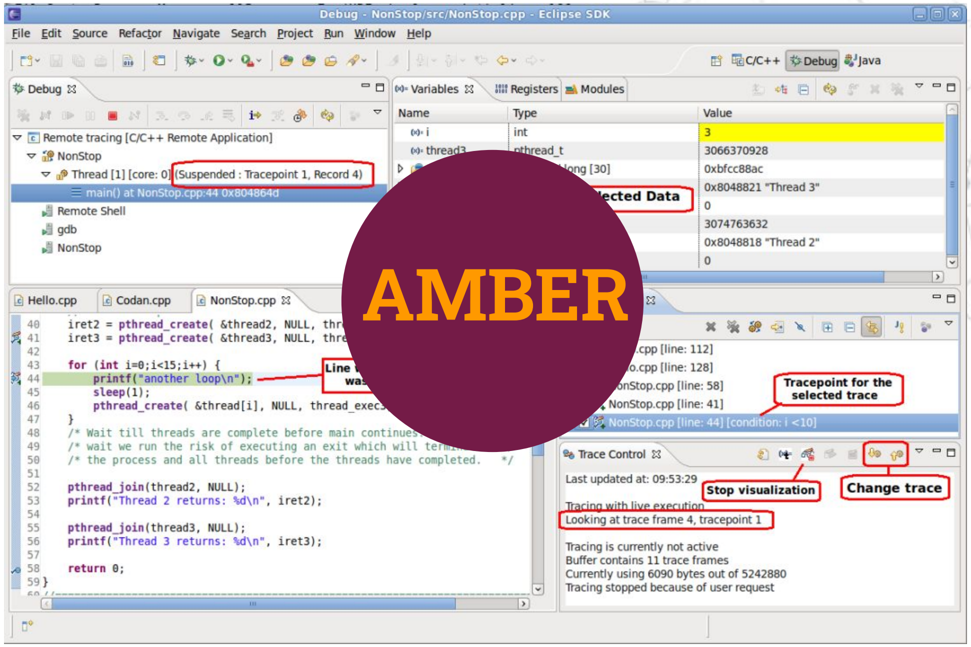
Problems in current systems:

- ❑ Little feedback
- ❑ No runtime debugging

printf() vs gdb



```
satapouch@satapouchPC ~/Stažené $ cat boot.log
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Running /scripts/local-premount ... done.
Begin: Running /scripts/local-bottom ... done.
done.
Begin: Running /scripts/init-bottom ... done.
 * Stopping load modules from /etc/modules                              [ OK ]
 * Stopping cold plug devices                                           [ OK ]
 * Stopping log initial device creation                                 [ OK ]
 * Starting enable remaining boot-time encrypted block devices          [ OK ]
 * Stopping Read required files in advance                              [ OK ]
 * Starting Mount filesystems on boot                                   [ OK ]
 * Stopping Track if upstart is running in a container                  [ OK ]
 * Starting Signal sysvinit that virtual filesystems are mounted        [ OK ]
 * Starting Signal sysvinit that virtual filesystems are mounted        [ OK ]
 * Starting Signal sysvinit that remote filesystems are mounted         [ OK ]
 * Starting Signal sysvinit that the rootfs is mounted                  [ OK ]
 * Starting Clean /tmp directory                                        [ OK ]
 * Stopping Clean /tmp directory                                        [ OK ]
 * Starting SMB/CIFS File Server                                        [ OK ]
 * Starting Signal sysvinit that local filesystems are mounted          [ OK ]
 * Starting restore software rfkill state                               [ OK ]
 * Starting configure network device security                           [ OK ]
 * Stopping restore software rfkill state                               [ OK ]
 * Stopping Mount filesystems on boot                                   [ OK ]
 * Starting flush early job output to logs                              [ OK ]
```



Remote tracing [C/C++ Remote Application]
NonStop
Thread [1] [core: 0] (Suspended : Tracepoint 1, Record 4)
main() at NonStop.cpp:44 0x804864d
Remote Shell
gdb
NonStop

Name	Type	Value
i	int	3
thread3	pthread_t	3066370928
	long [30]	0xbfcc88ac
		0x8048821 "Thread 3"
		0
		3074763632
		0x8048818 "Thread 2"
		0

```
    iret2 = pthread_create( &thread2, NULL, thr
    iret3 = pthread_create( &thread3, NULL, thre

    for (int i=0;i<15;i++) {
        printf("another loop\n");
        sleep(1);
        pthread_create( &thread[i], NULL, thread_exec3
    }
    /* Wait till threads are complete before main continues.
    /* wait we run the risk of executing an exit which will termi
    /* the process and all threads before the threads have completed.   */

    pthread_join(thread2, NULL);
    printf("Thread 2 returns: %d\n", iret2);

    pthread_join(thread3, NULL);
    printf("Thread 3 returns: %d\n", iret3);

    return 0;
}
```

.cpp [line: 112]
o.cpp [line: 128]
onStop.cpp [line: 58]
NonStop.cpp [line: 41]
NonStop.cpp [line: 44] [condition: i <10]

Last updated at: 09:53:29

Tracing with live execution
Looking at trace frame 4, tracepoint 1

Tracing is currently not active
Buffer contains 11 trace frames
Currently using 6090 bytes out of 5242880
Tracing stopped because of user request

❏ Hard to understand and track

❏ Real-time interaction

❏ Bug-related information

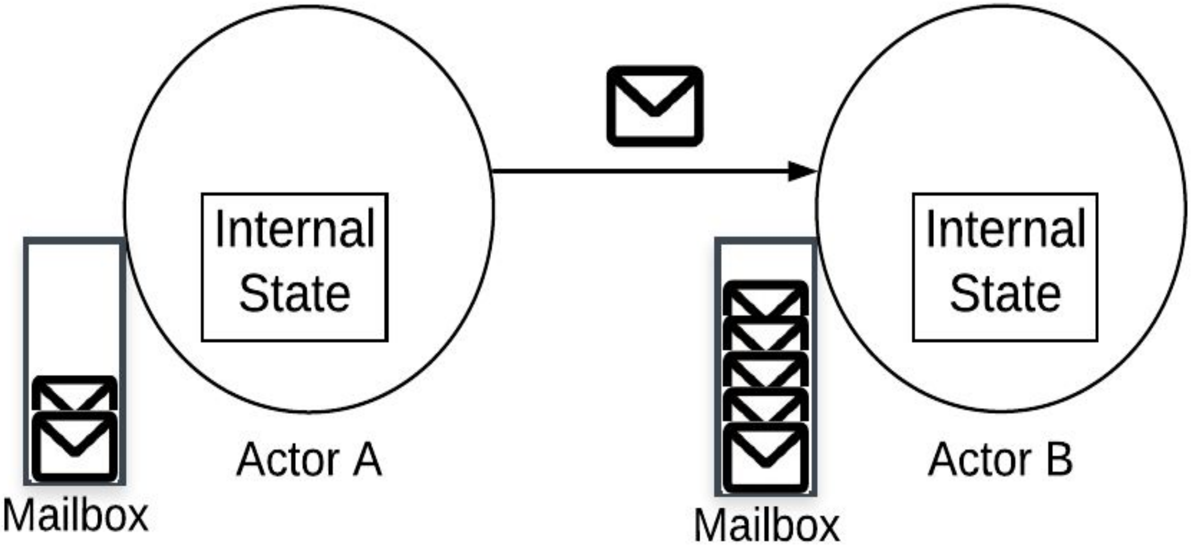
❏ Easy to understand and track

Why not Spark?

- 2-way communication between driver and executors needed for interacting with a running workflow

- Not supported in Spark to ensure deterministic computation

Actor Model


Internal State
Internal State
Actor A
Actor B
Mailbox
Mailbox

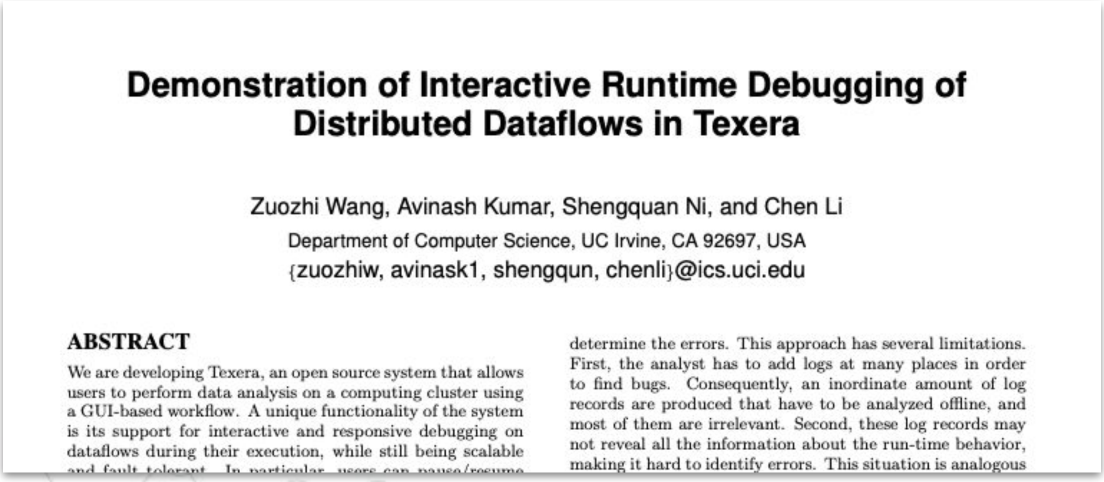
Why actor model?

- ❑ Parallel
- ❑ Simplifies concurrency control, no shared memory
- ❑ Easy to have different types of messages - data, control
- ❑ Good cluster implementations are available now
    - Microsoft Orleans
    - Akka
    - C++ Actor Framework (CAF)
    - ...

Demo Paper:

Demonstration of Interactive Runtime Debugging of Distributed Dataflows in Texera.

- Zuozhi Wang, Avinash Kumar, Shengquan Ni, Chen Li. VLDB 2020



Demonstration of Interactive Runtime Debugging of Distributed Dataflows in Texera

Zuozhi Wang, Avinash Kumar, Shengquan Ni, and Chen Li
Department of Computer Science, UC Irvine, CA 92697, USA
{zuozhiw, avinask1, shengqun, chenli}@ics.uci.edu

ABSTRACT

We are developing Texera, an open source system that allows users to perform data analysis on a computing cluster using a GUI-based workflow. A unique functionality of the system is its support for interactive and responsive debugging on dataflows during their execution, while still being scalable and fault tolerant. In particular, users can pause/resume

determine the errors. This approach has several limitations. First, the analyst has to add logs at many places in order to find bugs. Consequently, an inordinate amount of log records are produced that have to be analyzed offline, and most of them are irrelevant. Second, these log records may not reveal all the information about the run-time behavior, making it hard to identify errors. This situation is analogous

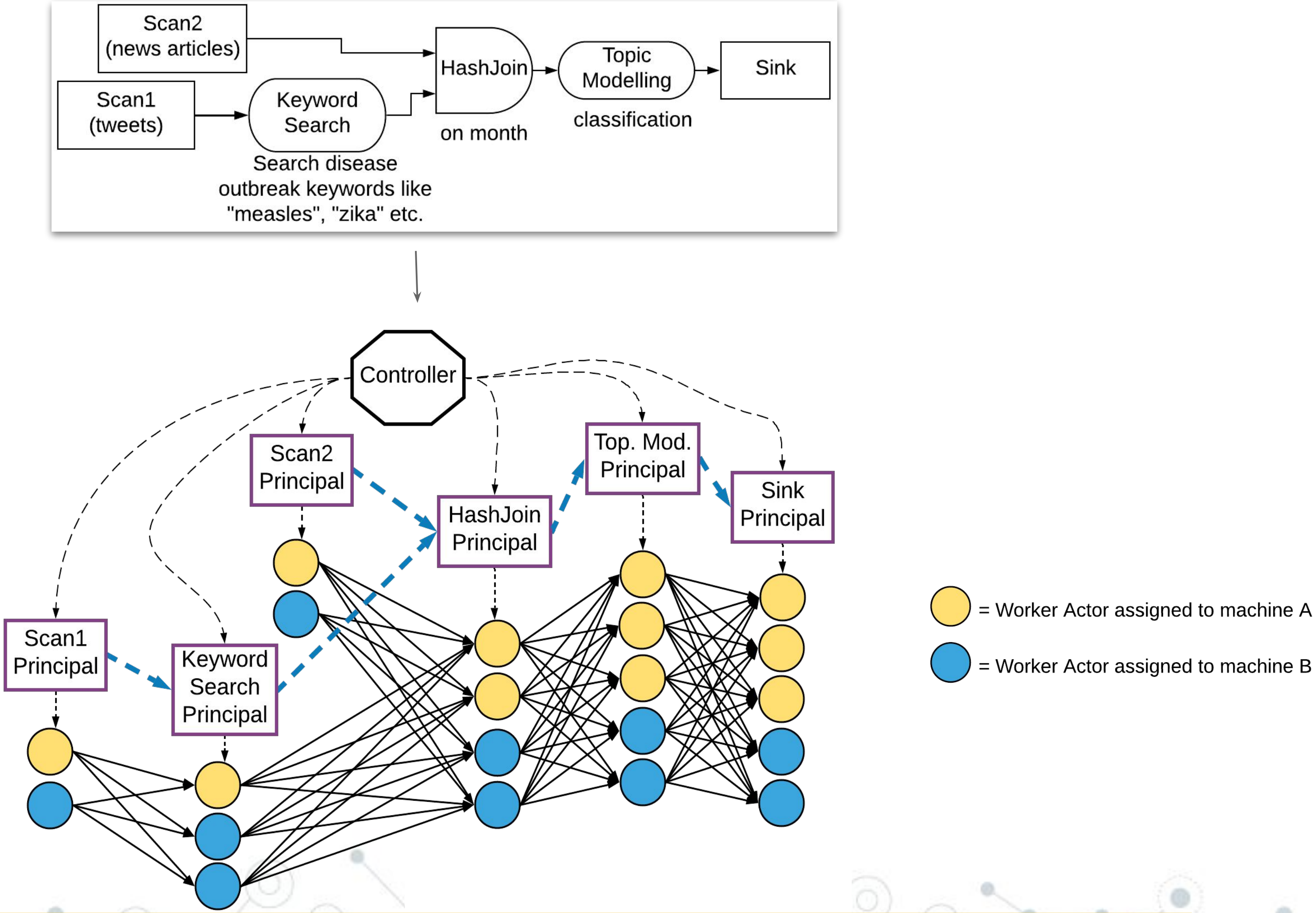
[29d] [Demo] Visualization, Explonation, Cleaning

[39d] [Demo] Visualization, Explonation, Cleaning

2.

Amber Overview

Operator DAG -> Actor DAG



Debugging a long-running workflow: An example

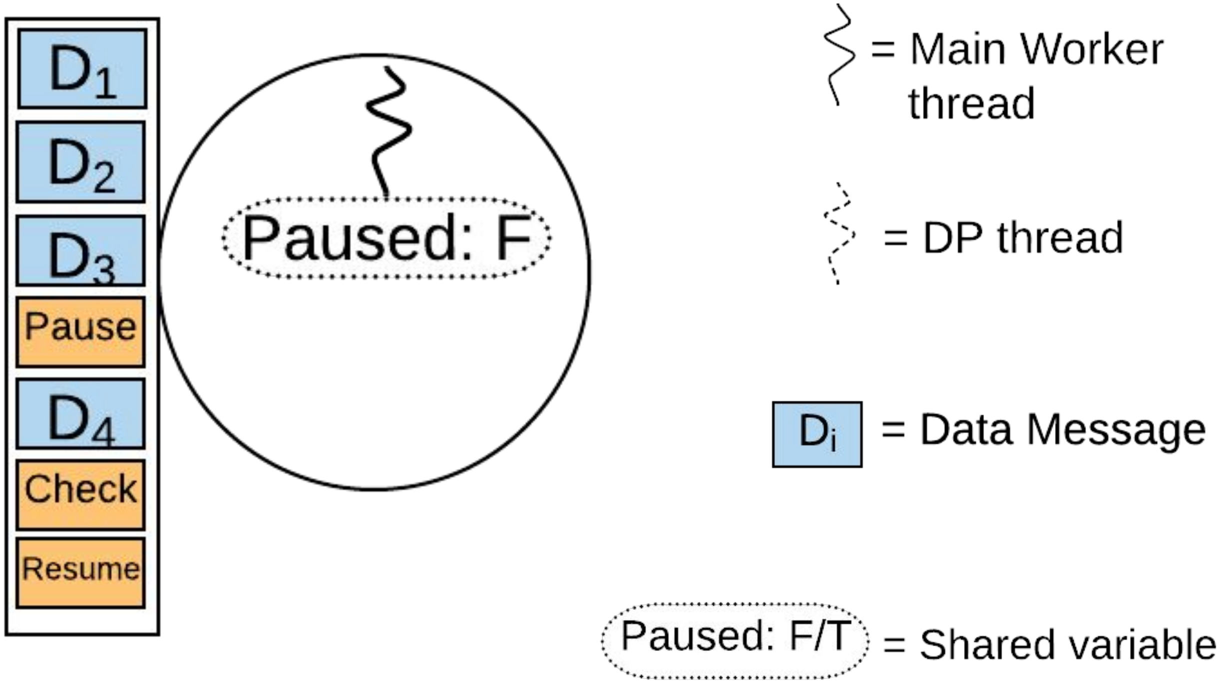
- Pause
- Investigate state of operators
- Modify Logic of operators
- Resume

Instantaneous processing of control messages

CHALLENGE

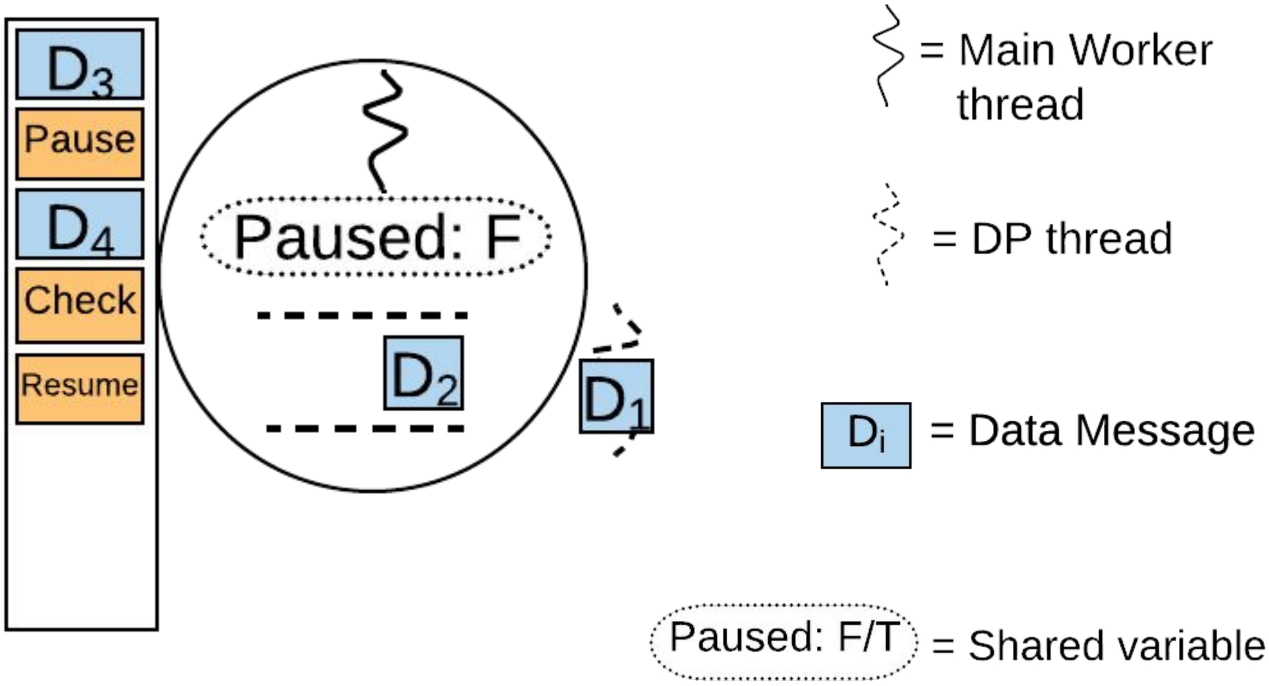
Expedited Control message processing

Messages arrive at Actor’s mailbox.



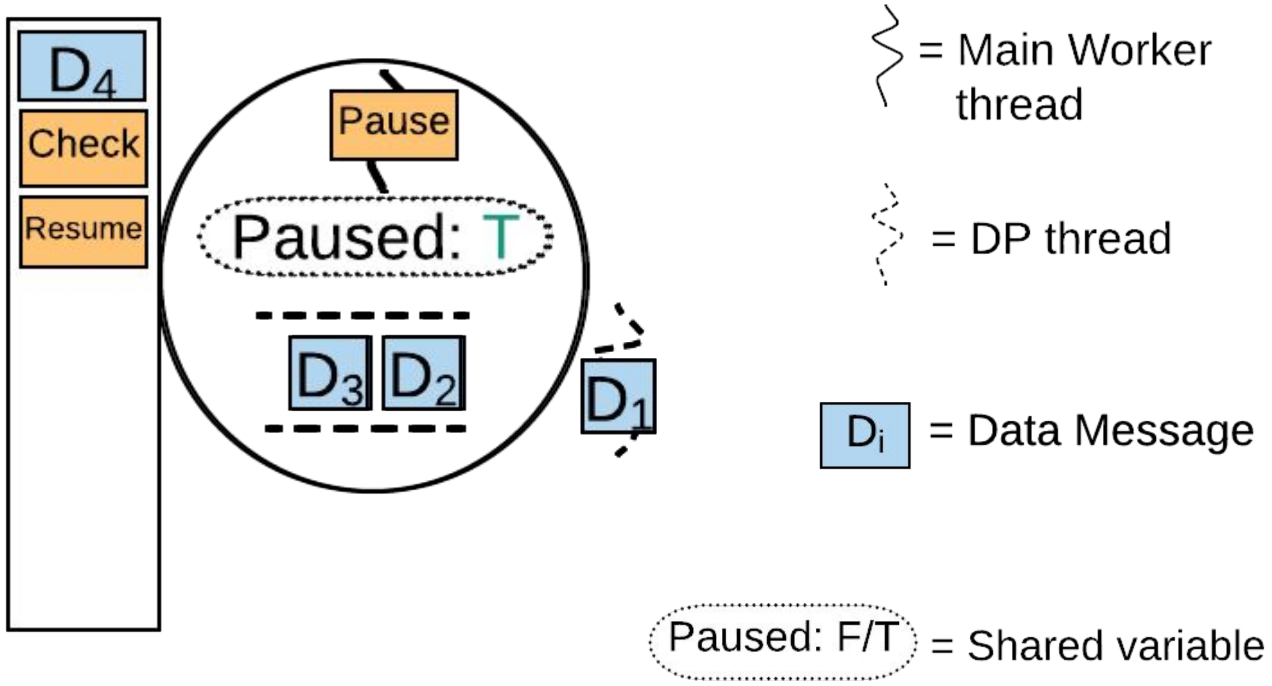
Expedited Control message processing

DP thread created.



Expedited Control message processing

Main Worker thread is available for control messages.



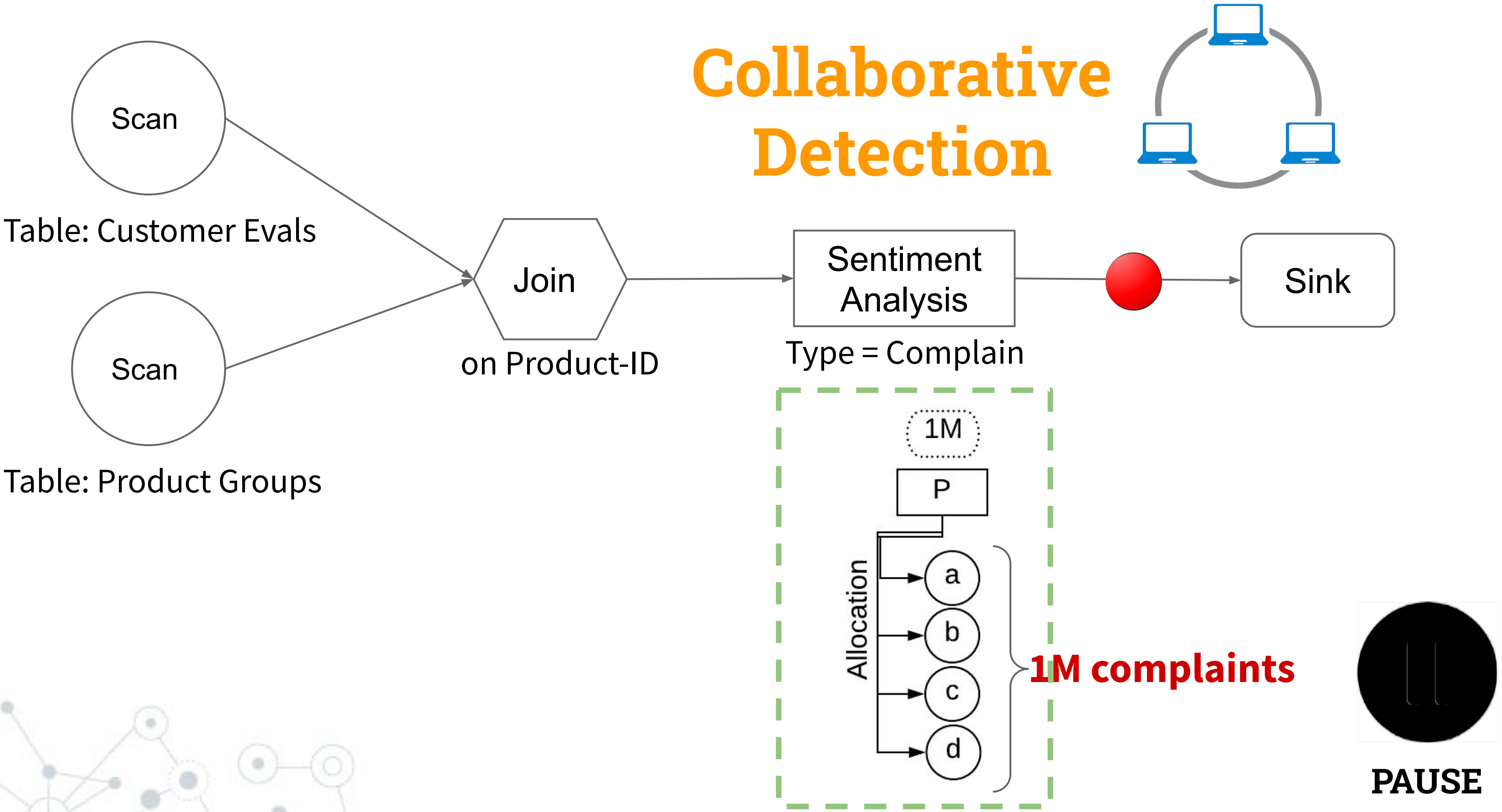
4.

Conditional Breakpoints

Let the workflow know when it should pause itself

Example

Pause the workflow if : # complaints > 1,000,000/product



Collaborative Detection

1M complaints

PAUSE

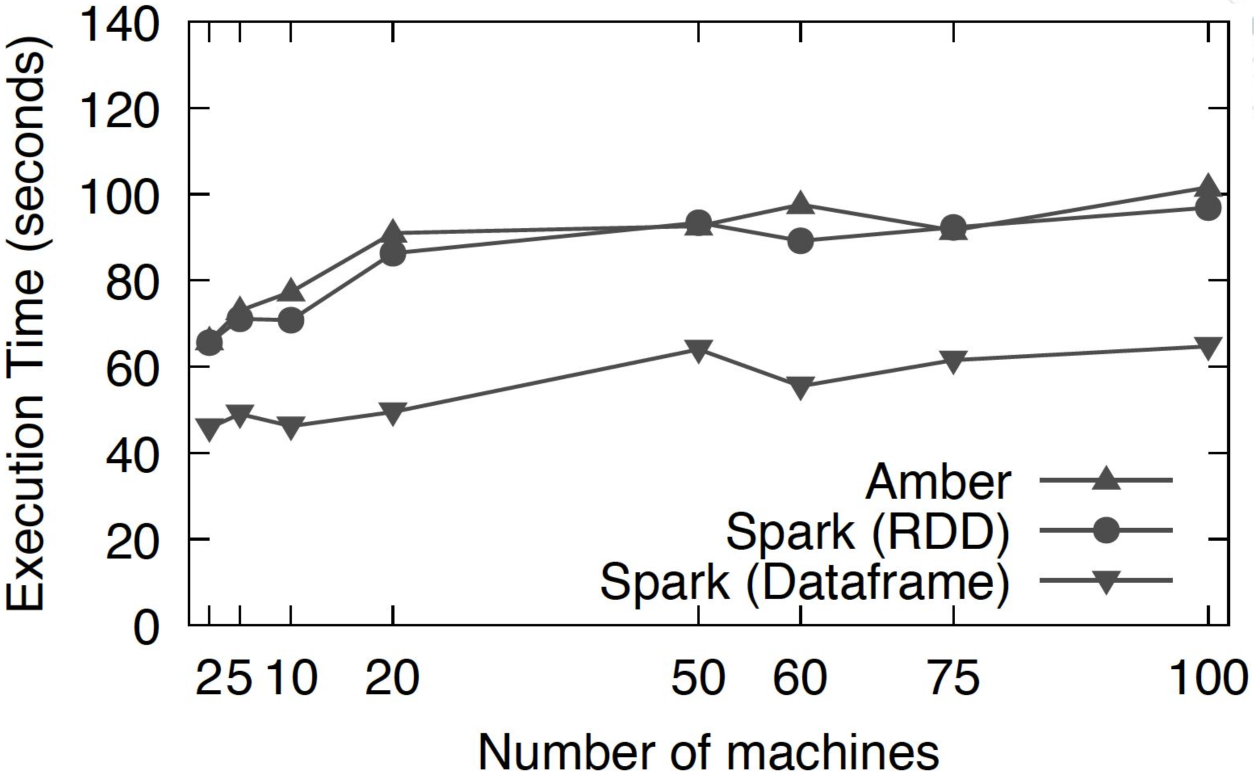
Fault Tolerance

- ❑ Interactions makes computation non-deterministic
- ❑ Fault tolerance in Amber = Same result **AND** Same state (Eg: Pause).
- ❑ Log order of control messages w.r.t data messages
- ❑ Replay both data and control messages

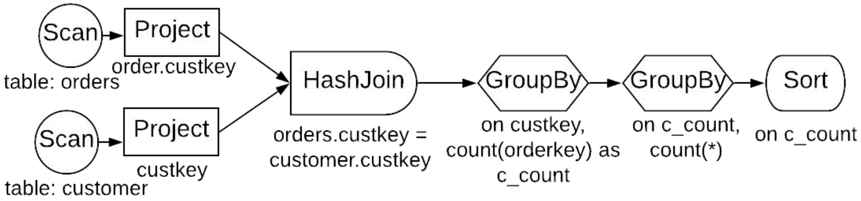
4.

Experiments

Scaleup on TPC-H Queries

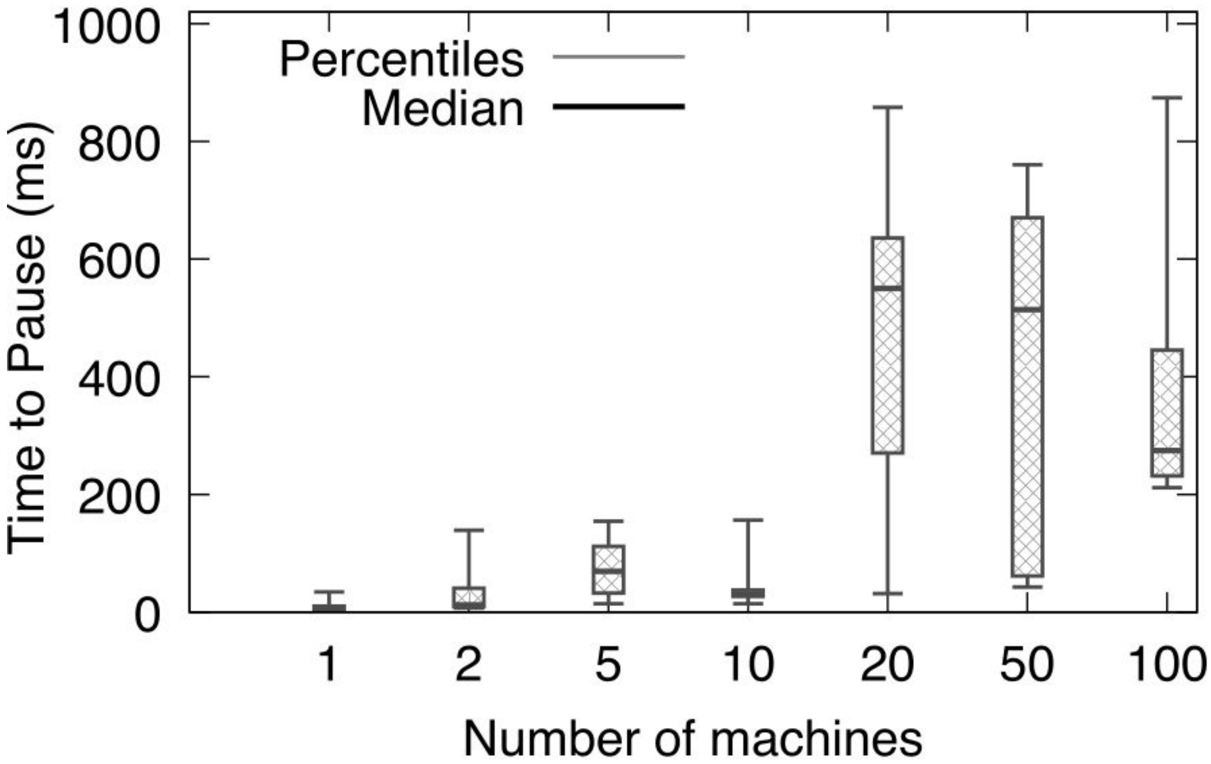


- 2GB to 1TB TPC-H Dataset
- Upto 100 machines
- Comparable to Spark

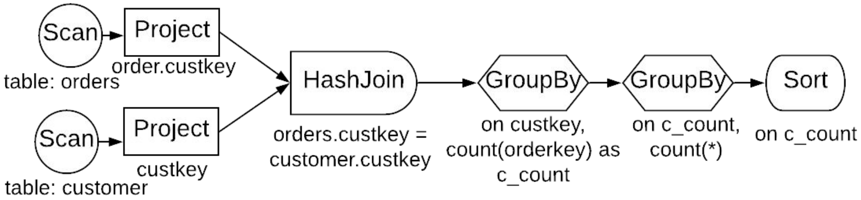


Simplified TPC-H Q13

Latency of control messages

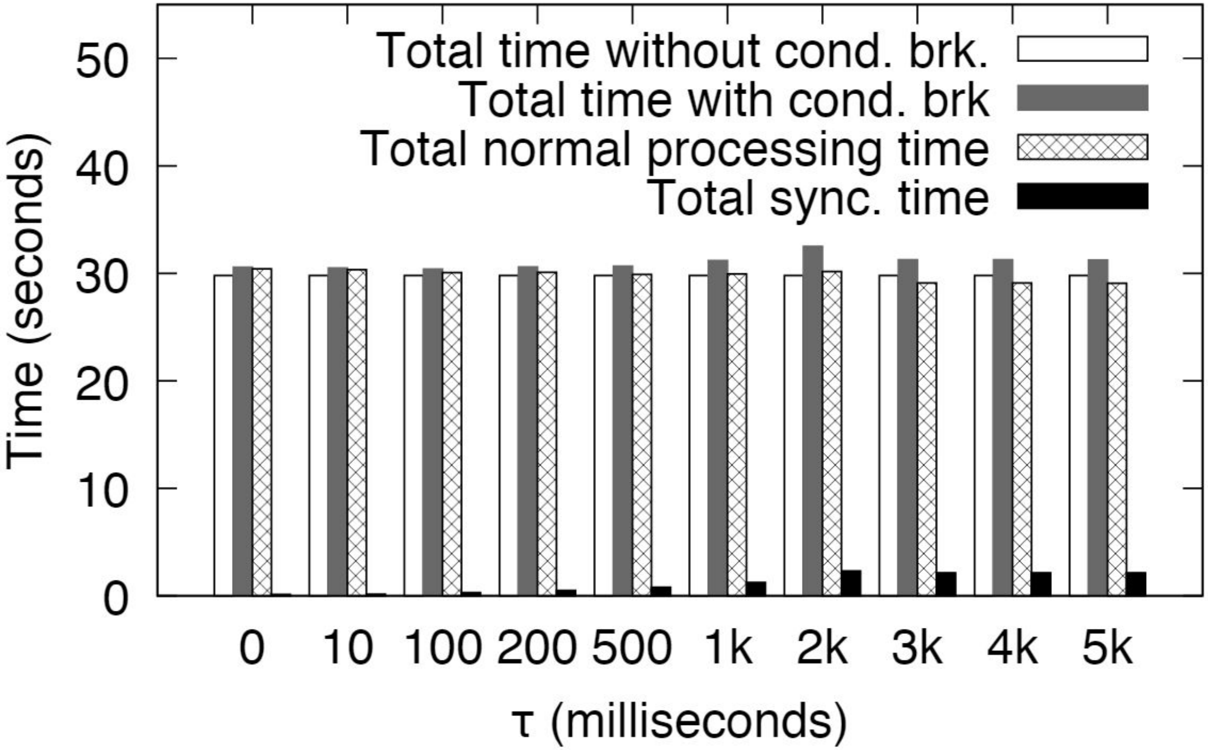


- ❑ Under 1 second
- ❑ Resume also in milliseconds
- ❑ Tested on 100 machines and 1TB data

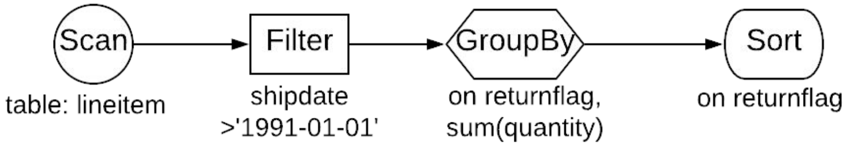


Simplified TPC-H Q13

Overhead of breakpoints



- ❏ Processing cluster: 10 machines
- ❏ Input: 119M tuples
- ❏ Pause Condition: Filter output 100M tuples



Simplified TPC-H Q1

Conclusions

- ❑ Amber: a parallel data flow engine supporting debuggability
- ❑ Based on the actor model
- ❑ Scalable
- ❑ Efficient pause/resume/conditional breakpoints

Amber : A Debuggable Dataflow System Based on the Actor Model

Avinash Kumar

Zuozhi Wang

Shengquan Ni

Chen Li

UCI University of California, Irvine